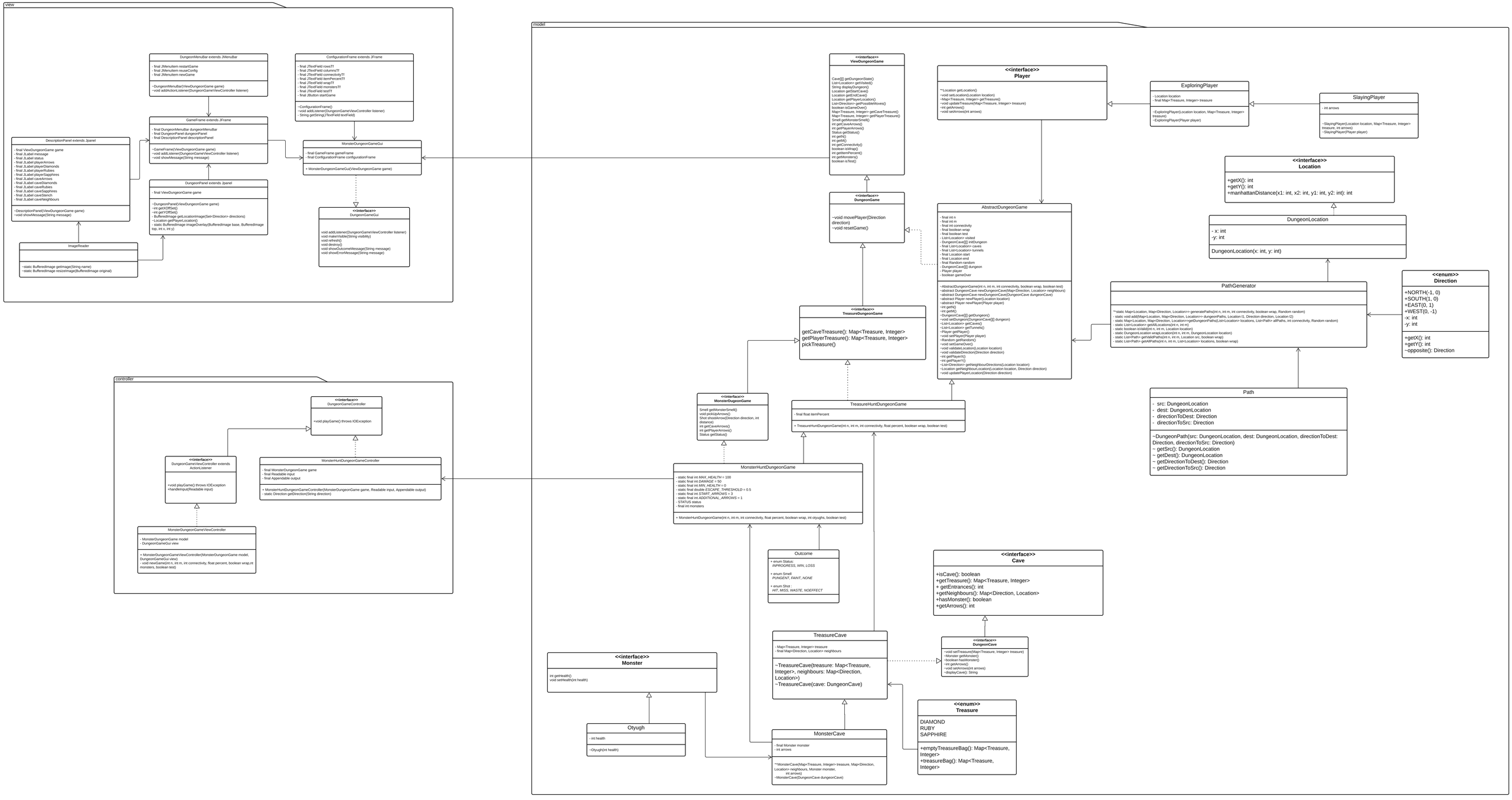
view
DungeonMenuBar extends JMenuBar
- final JMenuItem restartGame
- final JMenuItem reuseConfig
- final JMenuItem newGame
~DungeonMenuBar(ViewDungeonGame game)
~void addActionListener(DungeonGameViewController listener)
ConfigurationFrame extends JFrame
- final JTextField rowsTf
- final JTextField columnsTf
- final JTextField connectivityTf
- final JTextField itemPercentTf
- final JTextField wrapTf
- final JTextField monstersTf
- final JTextField testTf
- final JButton startGame
~ConfigurationFrame()
~void addListener(DungeonGameViewController listener)
- String getString(JTextField textField)
GameFrame extends JFrame
- final DungeonMenuBar dungeonMenuBar
- final DungeonPanel dungeonPanel
- final DescriptionPanel descriptionPanel
~GameFrame(ViewDungeonGame game)
~void addListener(DungeonGameViewController listener)
~void showMessage(String message)
DescriptionPanel extends Jpanel
- final ViewDungeonGame game
- final JLabel message
- final JLabel status
- final JLabel playerArrows
- final JLabel playerDiamonds
- final JLabel playerRubies
- final JLabel playerSapphires
- final JLabel caveArrows
- final JLabel caveDiamonds
- final JLabel caveRubies
- final JLabel caveSapphires
- final JLabel caveStench
- final JLabel caveNeighbours
~DescriptionPanel(ViewDungeonGame game)
~void showMessage(String message)
MonsterDungeonGameGui
- final GameFrame gameFrame
- final ConfigurationFrame configurationFrame
+ MonsterDungeonGameGui(ViewDungeonGame game)
DungeonPanel extends Jpanel
- final ViewDungeonGame game
~DungeonPanel(ViewDungeonGame game)
~int getXOffSet()
~int getYOffSet()
- BufferedImage getLocationImage(Set<Direction> directions)
~Location getPlayerLocation()
- static BufferedImage imageOverlay(BufferedImage base, BufferedImage top, int x, int y)
<<interface>>
DungeonGameGui
void addListener(DungeonGameViewController listener)
void makeVisible(String visibility)
void refresh()
void destroy()
void showOutcomeMessage(String message)
void showErrorMessage(String message)
ImageReader
~static BufferedImage getImage(String name)
~static BufferedImage resizeImage(BufferedImage original)
model
<<interface>>
ViewDungeonGame
Cave[][] getDungeonState()
List<Location> getValidat()
String displayDungeon()
Location getStartCave()
Location getEndCave()
Location getPlayerLocation()
List<Direction> getPossibleMoves()
boolean isGameOver()
Map<Treasure, Integer> getCaveTreasure()
Map<Treasure, Integer> getPlayerTreasure()
Smell getMonsterSmell()
int getCaveArrows()
int getPlayerArrows()
Status getStatus()
int getN()
int getM()
int getConnectivity()
boolean isWrap()
int gettemPercent()
int getMonsters()
boolean isTest()
<<interface>>
Player
~~Location getLocation()
~void setLocation(Location location)
~Map<Treasure, Integer> getTreasure()
~void updateTreasure(Map<Treasure, Integer> treasure)
~int getArrows()
~void setArrows(int arrows)
ExploringPlayer
- Location location
- final Map<Treasure, Integer> treasure
~ExploringPlayer(Location location, Map<Treasure, Integer> treasure)
~ExploringPlayer(Player player)
SlayingPlayer
- int arrows
~SlayingPlayer(Location location, Map<Treasure, Integer> treasure, int arrows)
~SlayingPlayer(Player player)
<<interface>>
DungeonGame
~void movePlayer(Direction direction)
~void resetGame()
AbstractDungeonGame
- final int n
- final int m
- final int connectivity
- final boolean wrap
- final boolean test
- List<Location> visited
- DungeonCave[][] initDungeon
- final List<Location> caves
- final List<Location> tunnels
- final Location start
- final Location end
- final Random random
- DungeonCave[][] dungeon
- Player player
- boolean gameOver
~AbstractDungeonGame(int n, int m, int connectivity, boolean wrap, boolean test)
~abstract DungeonCave newDungeonCave(Map<Direction, Location> neighbours)
~abstract DungeonCave newDungeonCave(DungeonCave dungeonCave)
~abstract Player newPlayer(Location location)
~abstract Player newPlayer(Player player)
~int getN()
~int getM()
~DungeonCave[][] getDungeon()
~void setDungeon(DungeonCave[][] dungeon)
~List<Location> getCaves()
~List<Location> getTunnels()
~Player getPlayer()
~void setPlayer(Player player)
~Random getRandom()
~void setGameOver()
~void validateLocation(Location location)
~void validateDirection(Direction direction)
~int getPlayerX()
~int getPlayerY()
~List<Direction> getNeighbourDirections(Location location)
~Location getNeighbourLocation(Location location, Direction direction)
~void updatePlayerLocation(Direction direction)
<<interface>>
Location
+getX(): int
+getY(): int
+manhattanDistance(x1: int, x2: int, y1: int, y2: int): int
DungeonLocation
- x: int
-y: int
DungeonLocation(x: int, y: int)
<<interface>>
TreasureDungeonGame
getCaveTreasure(): Map<Treasure, Integer>
getPlayerTreasure(): Map<Treasure, Integer>
pickTreasure()
PathGenerator
~~static Map<Location, Map<Direction, Location>> generatePaths(int n, int m, int connectivity, boolean wrap, Random random)
- static void add(Map<Location, Map<Direction, Location>> dungeonPaths, Location l1, Direction direction, Location l2)
- static Map<Location, Map<Direction, Location>>getDungeonPaths(List<Location> locations, List<Path> allPaths, int connectivity, Random random)
- static List<Location> getAllLocations(int n, int m)
- static boolean isValid(int n, int m, Location location)
- static DungeonLocation wrapLocation(int n, int m, DungeonLocation location)
- static List<Path> getValidPaths(int n, int m, Location src, boolean wrap)
- static List<Path> getAllPaths(int n, int m, List<Location> locations, boolean wrap)
<<enum>>
Direction
+NORTH(-1, 0)
+SOUTH(1, 0)
+EAST(0, 1)
+WEST(0, -1)
-x: int
-y: int
+getX(): int
+getY(): int
~opposite(): Direction
<<interface>>
MonsterDugeonGame
Smell getMonsterSmell()
void pickUpArrows()
Shot shootArrow(Direction direction, int distance)
int getCaveArrows()
int getPlayerArrows()
Status getStatus()
TreasureHuntDungeonGame
- final float itemPercent
+ TreasureHuntDungeonGame(int n, int m, int connectivity, float percent, boolean wrap, boolean test)
Path
- src: DungeonLocation
- dest: DungeonLocation
- directionToDest: Direction
- directionToSrc: Direction
~DungeonPath(src: DungeonLocation, dest: DungeonLocation, directionToDest: Direction, directionToSrc: Direction)
~ getSrc(): DungeonLocation
~ getDest(): DungeonLocation
~ getDirectionToDest(): Direction
~ getDirectionToSrc(): Direction
controller
<<interface>>
DungeonGameController
+void playGame() throws IOException
<<interface>>
DungeonGameViewController extends ActionListener
+void playGame() throws IOException
+handleInput(Readable input)
MonsterHuntDungeonGameController
- final MonsterDungeonGame game
- final Readable input
- final Appendable output
+ MonsterHuntDungeonGameController(MonsterDungeonGame game, Readable input, Appendable output)
- static Direction getDirection(String direction)
MonsterDungeonGameViewController
- MonsterDungeonGame model
- DungeonGameGui view
+ MonsterDungeonGameViewController(MonsterDungeonGame model, DungeonGameGui view)
- void newGame(int n, int m, int connectivity, float percent, boolean wrap,int monsters, boolean test)
MonsterHuntDungeonGame
- static final int MAX_HEALTH = 100
- static final int DAMAGE = 50
- static final int MIN_HEALTH = 0
- static final double ESCAPE_THRESHOLD = 0.5
- static final int START_ARROWS = 3
- static final int ADDITIONAL_ARROWS = 1
- STATUS status
- final int monsters
+ MonsterHuntDungeonGame(int n, int m, int connectivity, float percent, boolean wrap, int otyughs, boolean test)
Outcome
+ enum Status:
INPROGRESS, WIN, LOSS
+ enum Smell :
PUNGENT, FAINT, NONE
+ enum Shot :
HIT, MISS, WASTE, NOEFFECT
<<interface>>
Cave
+isCave(): boolean
+getTreasure(): Map<Treasure, Integer>
+ getEntrances(): int
+getNeighbours(): Map<Direction, Location>
+hasMonster(): boolean
+getArrows(): int
TreasureCave
- Map<Treasure, Integer> treasure
- final Map<Direction, Location> neighbours
~TreasureCave(treasure: Map<Treasure, Integer>, neighbours: Map<Direction, Location>)
~TreasureCave(cave: DungeonCave)
<<interface>>
DungeonCave
~void setTreasure(Map<Treasure, Integer> treasure)
~Monster getMonster()
~boolean hasMonster()
~int getArrows()
~void setArrows(int arrows)
~displayCave(): String
<<interface>>
Monster
int getHealth()
void setHealth(int health)
Otyugh
- int health
~Otyugh(int health)
MonsterCave
- final Monster monster
- int arrows
~~MonsterCave(Map<Treasure, Integer> treasure, Map<Direction, Location>> neighbours, Monster monster, int arrows)
~MonsterCave(DungeonCave dungeonCave)
<<enum>>
Treasure
DIAMOND
RUBY
SAPPHIRE
+emptyTreasureBag(): Map<Treasure, Integer>
+treasureBag(): Map<Treasure, Integer>